Numpy Library: Linear Algebra

1. **Write a python program to find rank, determinant, and trace of an array.**

```
import numpy as np

# Input the matrix (replace this with your own matrix)
matrix = np.array([[1, 2, 3],
                   [4, 5, 6],
                   [7, 8, 9]])

# Calculate the rank of the matrix
rank = np.linalg.matrix_rank(matrix)

# Calculate the determinant of the matrix
determinant = np.linalg.det(matrix)

# Calculate the trace of the matrix
# FYI- the trace of the matrix means its  the sum of the elements on its main diagonal, which
#runs from the top left to the bottom right of the matrix.

trace = np.trace(matrix)

# Print the results
print("Matrix:")
print(matrix)
print(f"Rank: {rank}")
print(f"Determinant: {determinant}")
print(f"Trace: {trace}")
```

2. **Write a python program to find eigenvalues of matrices.**

```
import numpy as np

# Input the matrix (replace this with your own matrix)
matrix = np.array([[1, 2], [2, 3]])

# Calculate the eigenvalues of the matrix
eigenvalues = np.linalg.eigvals(matrix)

# Print the eigenvalues
print("Matrix:")
print(matrix)
print("Eigenvalues:")
print(eigenvalues)
```

3. **Write a python program to find matrix and vector products (dot, inner, outer, product), matrix exponentiation.**

```
import numpy as np

# Create matrices and vectors (replace these with your own)
matrix_A = np.array([[1, 2], [3, 4]])
matrix_B = np.array([[5, 6], [7, 8]])
vector_x = np.array([1, 2])
vector_y = np.array([3, 4])

# Dot product of two vectors
dot_product = np.dot(vector_x, vector_y)

# Inner product (dot product) of two matrices
inner_product = np.inner(matrix_A, matrix_B)

# Outer product of two vectors
outer_product = np.outer(vector_x, vector_y)

# Matrix product (matrix multiplication)
matrix_product = np.matmul(matrix_A, matrix_B)

# Matrix exponentiation
exponent = 2  # Replace with the desired exponent
matrix_exponential = np.linalg.matrix_power(matrix_A, exponent)

# Print the results
print("Vector x:", vector_x)
print("Vector y:", vector_y)
print("Matrix A:")
print(matrix_A)
print("Matrix B:")
print(matrix_B)
print("Dot product of x and y:", dot_product)
print("Inner product of A and B:")
print(inner_product)
print("Outer product of x and y:")
print(outer_product)
print("Matrix product of A and B:")
print(matrix_product)
print(f"Matrix A raised to the power {exponent}:")
print(matrix_exponential)
```

4. **Write a python program to solve a linear matrix equation, or system of linear scalar equations.**

```
import numpy as np

# Define the coefficients matrix (A) and the right-hand side vector (b)
# Replace these with your own values
A = np.array([[2, 1],
              [1, 3]])

b = np.array([3, 9])

# Solve the linear system of equations
solution = np.linalg.solve(A, b)

# Print the solution
print("Coefficient matrix (A):")
print(A)
print("Right-hand side vector (b):")
print(b)
print("Solution vector (x):")
print(solution)
```

Pandas Library

1. **Write a python program to implement Pandas Series with labels.**

```
import pandas as pd

# Create a pandas Series with labels
data = {'A': 10, 'B': 20, 'C': 30, 'D': 40}
my_series = pd.Series(data)

# Print the Series
print("Pandas Series:")
print(my_series)

# Accessing elements by label
print("\nAccessing elements by label:")
print("Value at label 'A':", my_series['A'])
print("Value at label 'C':", my_series['C'])

# Accessing elements by position
print("\nAccessing elements by position:")
```

```
print("The first element:", my_series[0])
print("The second element:", my_series[1])

# Arithmetic operations on the Series
print("\nArithmetic operations on the Series:")
result = my_series * 2
print("Series multiplied by 2:")
print(result)

# Filtering elements
print("\nFiltering elements:")
filtered_series = my_series[my_series > 20]
print("Elements greater than 20:")
print(filtered_series)
```

2. **Create a Pandas Series from a dictionary.**

```
import pandas as pd

# Create a dictionary
data_dict = {'apple': 3, 'banana': 2, 'cherry': 5, 'date': 1}

# Create a pandas Series from the dictionary
fruits_series = pd.Series(data_dict)

# Print the Series
print(fruits_series)
```

3. **Creating a Pandas DataFrame.**

```
import pandas as pd

# Create a dictionary of data
data = {
   'Name': ['Alice', 'Bob', 'Charlie', 'David'],
   'Age': [25, 30, 35, 40],
   'City': ['New York', 'San Francisco', 'Los Angeles', 'Chicago']
}

# Create a Pandas DataFrame from the dictionary
df = pd.DataFrame(data)

# Print the DataFrame
print(df)
```

In this program:

- We create a dictionary called data, where each key represents a column name, and the corresponding values are lists containing the data for each column.
- We use the pd.DataFrame(data) constructor to create a Pandas DataFrame from the dictionary.
- Finally, we print the DataFrame df, which will display the data in a tabular format.

4. Write a program which make use of following Panda's methods.
   a. describe ()
   b. head ()
   c. tail ()

```
import pandas as pd

# Create a sample DataFrame
data = {
   'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],
   'Age': [25, 30, 35, 40, 22],
   'City': ['New York', 'San Francisco', 'Los Angeles', 'Chicago', 'Miami']
}
df = pd.DataFrame(data)
# Use describe() to get summary statistics
print("DataFrame Statistics (describe()):")
print(df.describe())

# Use head() to display the first few rows (default is 5)
print("\nFirst 3 rows (head()):")
print(df.head(3))

# Use tail() to display the last few rows (default is 5)
print("\nLast 2 rows (tail()):")
print(df.tail(2))
```

In this program:

- We create a sample DataFrame df from a dictionary data.
- We use the describe() method to get summary statistics of the numerical columns in the DataFrame. It provides statistics like mean, standard deviation, minimum, maximum, and quartiles.
- We use the head(3) method to display the first 3 rows of the DataFrame. You can change the argument to show a different number of rows.

- We use the tail(2) method to display the last 2 rows of the DataFrame. You can change the argument to show a different number of rows.